An introduction to error-correcting codes

Digital communications – binary sequences

In digital communications information is often sent as a sequence of 1s and 0s. Such a sequence is called a *binary* sequence and the individual 1s and 0s are called *binary digits* or *bits*. One of the earliest such systems was that devised in 1870 by Emile Baudot, where each letter of the alphabet was represented by a sequence of five bits. A later variant of this system called ITA2 (International Telegraph Alphabet No 2), informally referred to as *baudot*, is still in use on the amateur bands today for the RTTY mode (radio teletype).

A much more modern system called American Standard Code for Information Interchange (ASCII) was developed during the early 1960s and a revised version of ASCII is still widely used today in computer systems. In ASCII, each symbol (capital letter, lowercase letter, numeral, punctuation mark, plus various non-printable control characters) is assigned a decimal number in the range 0 to 127. Each of these decimal numbers may in turn be represented by a sequence of seven bits. For example, a capital letter J is assigned decimal number 74, which is 64 + 8 + 2 and hence represented by the seven-bit sequence 1001010. Having seven bits, rather than just five, gives four times as many binary sequences and allows a much larger set of symbols to be represented.

Binary data may be sent over a communication channel by a variety of means. For example, in amateur packet radio, data is sent using an FM signal modulated with a 1.2kHz audio tone for 1 and a 2.2kHz audio tone for 0. In practice, no communication channel is perfect and noise or interference may cause a 1 that is sent to be received as a 0, or vice versa. Clearly such errors compromise the integrity of the data. For example, suppose the seven-bit ASCII binary sequence 1001010 representing the capital letter J is sent and the final bit is received as a 1. The received sequence, 1001011, represents the decimal number 75 – assigned to the capital letter K. Hence, an operator's name sent as JIM will be received as KIM.

The purpose of error correcting codes is to enable recognition of such errors when they occur and, if possible, to correct them. Error correcting codes have been used in the amateur radio community for several decades. An early example is the data mode Amateur Teleprinting Over Radio (AMTOR), developed in the 1970s. AMTOR has a very basic error detection facility: each character is represented by a seven-bit sequence with precisely four bits equal to 1 and three bits equal to 0; any deviation from this pattern in the received sequence is recognised as an error. Many later data modes make use of error correction, including the WSJT suite, which was developed in the late 1990s; Opera, which is in widespread use on the LF bands; and several other HF data modes such as the very popular WSPR and JT65. For packet radio, an enhanced version of the AX25 packet protocol incorporating error correction, called FX25, was created in 2005.

The principal aim of this article is to make the theory of error correcting codes more widely known and understood within the amateur radio community, particularly those involved in the development of new data modes. Moreover, the subject is interesting and elegant in itself and underlies much of the digital technology that we all use on a daily basis, so should be of interest even to readers who do not use data modes.

Since its emergence in the 1950s, coding theory has grown into a deep and complex area of mathematics, and in this article we shall only scratch the surface of the myriad coding schemes now in use. We restrict our attention here to binary block codes, using the two symbols 0 and 1 transmitted in blocks of a given fixed length, since these are the most basic codes and the starting point for the development of more sophisticated coding schemes.

Repetition and parity check

The simplest scheme for detecting an error is a repetition code. For example, each bit of the sequence representing the capital letter J may be sent twice: 11 00 00 11 00 11 00. The two bit blocks 11 and 00 are called *code words*. Suppose that once again an error occurs in the final bit, so that the following sequence is received: 11 00 00 11 00 11 01. The recipient examines each block to check that it is a code word. The first six blocks pass the check but the final block, 01, is not a code word. Hence the error has been detected. However, there is no way of knowing whether the code word that was sent was 00 or 11.

For each bit in the original data, the repetition code requires that two bits be sent, and we say that this code has a *rate* of ½ = 0.50. It is desirable to achieve as high a rate as possible, and there are ways of improving upon the repetition code.

The *weight* of a binary sequence is the number of bits in the sequence that are equal to 1. For example, the sequence 1001010 has weight three, whilst 1001011 has weight four. A *parity check* code works by restricting code words to sequences of *even* weight. Beginning with the seven bit sequence representing J, we may pad the sequence with an additional 0 bit at the front and then break the eight bit sequence into four blocks of two bits: 01 00 10 10. We then append a third bit to each block, so that each resulting three bit block is of even weight: 011 000 101 101. This is called a *parity check* bit. Suppose that once again an error occurs in the final bit so that the following sequence is received: 011 000 101 100. The recipient sees that the final block, 100, has odd weight, and hence the error has been detected. Once again, there is no way of knowing which of the bits was erroneous: the code word sent may have been 000, 110 or 101.

The number of bits in the code words for a code is called the *length* of the code, denoted *n*. The number of data bits that are encoded in each code word is called the *dimension* of the code, denoted *k*. The rate of the code is given by *k*/*n*. The parity check code above has length 3 and dimension 2. Hence the code has rate $^2/_3 = 0.67$, which is an improvement over the rate of the repetition code. Further improvement of the rate may be made by lengthening the blocks to which the parity check bit is attached. We may break the eight bit sequence above into two blocks of four bits: 0100 1010. A parity check bit is then appended to each block to give 01001 10100. This code has length 5 and dimension 4, and so the rate is $^4/_5 = 0.80$. However, the improvement in rate has to be paid for in loss of information about the error that has occurred. For example, if the block 10101 is received then we know that an error has occurred. But, assuming that only a single error has occurred, the code word sent may have been 00101, 11101, 10001, 10111 or 10100.

Arithmetic modulo 2 & linear codes

In arithmetic modulo 2, as in binary, we work with a pair of symbols: 0 and 1. We have 0 + 0 = 0, 0 + 1 = 1 and 1 + 0 = 0. However, whereas in binary arithmetic we have 1 + 1 = 10, in modulo 2 arithmetic no bit is carried to the next place so that we have 1 + 1 = 0. We may extend modulo 2

arithmetic from pairs of single bits to pairs of sequences of bits of equal length, by adding bits in the corresponding positions. For example 10101 + 01110 = 11011. In a similar way, we may add together three or more sequences of the same length. For example, 11001 + 01101 + 00011 = 10111.

A *linear code* is one in which the sum of any two code words is itself a code word. For example, in the parity check code of length five from the previous section there are sixteen code words:

Weight zero: 00000
Weight two: 11000, 10100, 10010, 10001, 01100, 01010, 01001, 00110, 00101, 00011
Weight four: 11110, 11101, 11011, 10111, 01111.

It is easily checked that the sum of any pair of these is also a code word. For example, 11101 + 01111 = 10010. In this article, we are concerned only with linear codes.

An important concept for linear codes is *Hamming distance*, named after Richard Hamming, who first introduced the concept in 1950. For a pair of binary sequences X and Y of equal length, the Hamming distance between X and Y is defined to be the number of bits in which X and Y differ. We write this as d(X, Y). For example, d(1110101, 1111010) = 4 since the two seven-bit sequences agree in the first three bits but differ in the remaining four. Notice that the weight of a sequence X, which we write as w(X), is its Hamming distance from the all-zero sequence of the same length: w(X) = d(X, 0). Notice also that we may find the Hamming distance between sequences X and Y by finding the weight of their sum: d(X, Y) = w(X + Y). For example,

d(1110101, 1111010)
= w(1110101 + 1111010)
= w(0001111)
= 4.

Examining again the parity check code of length five, we see that the Hamming distance between any pair of distinct code words is either 2 or 4. It follows that the minimum Hamming distance between any pair of distinct code words is 2. The minimum Hamming distance, denoted d, is an important parameter of a linear code as it determines the number of errors in each block which can be corrected.

Single error correction – Hamming codes

In the second section we looked at repetition and parity check codes that enabled us to detect a single error in a sequence of bits. However, these codes did not enable us to *correct* an error that has occurred. The simplest way in which we can correct a single error in a block is to repeat each bit three times. For the seven bit sequence 1001010 representing J we get
111 000 000 111 000 111 000

Two code words, 000 and 111, are separated by Hamming distance 3.

Suppose once again that an error occurs in the final bit, so that the following sequence is received:
111 000 000 111 000 111 001

The first six blocks are code words, but the seventh block is not. Since 001 has a Hamming distance of two from the code word 111, but a Hamming distance of only one from the code word 000, if only a single error occurred then the code word sent must have been 000. Thus the triple repetition code allows us to correct a single error per block. However, the rate of this code is only ⅓ = 0.33, motivating the need for error correcting codes that have a higher rate.

An important class of linear codes known as *Hamming codes* (again named after Richard Hamming, their discoverer) allow the correction of a single bit error per block in a received sequence. The Hamming code of length 7 is defined by the rectangular array shown in **Figure 1**, known as a *generator matrix*.

Row 1	1	0	0	0	1	0	1
Row 2	0	1	0	0	1	1	1
Row 3	0	0	1	0	1	1	0
Row 4	0	0	0	1	0	1	1

FIGURE 1: Generator matrix for Hamming code of length 7.

0	0	0	0	>	0	0	0	0	0	0	0
0	0	0	1	>	0	0	0	1	0	1	1
0	0	1	0	>	0	0	1	0	1	1	0
0	0	1	1	>	0	0	1	1	1	0	1
0	1	0	0	>	0	1	0	0	1	1	1
0	1	0	1	>	0	1	0	1	1	0	0
0	1	1	0	>	0	1	1	0	0	0	1
0	1	1	1	>	0	1	1	1	0	1	0
1	0	0	0	>	1	0	0	0	1	0	1
1	0	0	1	>	1	0	0	1	1	1	0
1	0	1	0	>	1	0	1	0	0	1	1
1	0	1	1	>	1	0	1	1	0	0	0
1	1	0	0	>	1	1	0	0	0	1	0
1	1	0	1	>	1	1	0	1	0	0	1
1	1	1	0	>	1	1	1	0	1	0	0
1	1	1	1	>	1	1	1	1	1	1	1

FIGURE 2: Coding scheme for the Hamming code of length 7.

The number of rows in the array is the dimension of the code, denoted k. So for the Hamming code of length 7 we have $k = 4$. We encode four bit sequences of data by adding together the rows of the array that correspond to bits in the sequence that are equal to 1. For example, 1000 is encoded as the first row of the array, 1000101. The four bit sequence 0101 is encoded by adding together the second and fourth rows of the array to give 0100111 + 0001011 = 0101100. This leads to the coding scheme shown in **Figure 2**.

We notice that all of the non-zero code words are of weight 3, 4 or 7. For any code word X of non-zero weight w there are a pair of distinct code words separated by Hamming distance w, namely X itself and the all-zero word 0. Conversely, for any pair of distinct code words X and Y separated by Hamming distance w, there is a code word of weight w, namely X + Y. It follows that any pair of distinct code words in the Hamming code of length 7 are separated by Hamming distance 3, 4 or 7, and hence that the minimum Hamming distance of the code is $d = 3$.

Because the minimum Hamming distance is 3, if an error occurs in a single bit then the sequence that results will have Hamming distance one from the code word sent, but Hamming distance two or more from all other code words. Hence, on the assumption that only a single error has occurred, the error can be corrected. To implement error correction, we use a second rectangular array, known as the *parity check matrix*. This is shown in **Figure 3**.

Suppose that we break up the eight bit sequence representing J into two blocks of four bits as before: 0100 1010. This is encoded using the Hamming code to give two blocks of seven bits: 0100111 1010011

Let us suppose that errors occur in the seventh bit of the first block and in the second bit of the second block, so that the following sequence is received: 0100110 1110011

Because the first block has 1s in the second, fifth and sixth bit, we add together the second, fifth and sixth rows of the parity check matrix: 111 + 100 + 010 = 001

The three bit sequence 001 is called the *syndrome*. In this case the syndrome is precisely the seventh row of the parity check matrix, which tells us that the error occurred in the seventh bit.

The second block has 1s in the first, second, third, sixth and seventh bits. We add together the first, second, third, sixth and seventh rows of the parity check matrix: 101 + 111 + 110 + 010 + 001 = 111

This time the syndrome 111 is the second row of the parity check matrix, telling us that the error occurred in the second bit.

Dr Michael K Butler, G4OCR
m.k.butler@bolton.ac.uk

Row 1	1	0	1
Row 2	1	1	1
Row 3	1	1	0
Row 4	0	1	1
Row 5	1	0	0
Row 6	0	1	0
Row 7	0	0	1

FIGURE 3: The parity check matrix for the Hamming code of length 7.

Row 1	1	0	0	0	1	0	1	1
Row 2	0	1	0	0	1	1	1	0
Row 3	0	0	1	0	1	1	0	1
Row 4	0	0	0	1	0	1	1	1

FIGURE 4: Generator matrix for the extended Hamming code of length 8.

1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	1
0	1	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	1	1
0	0	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	1	0	0	0	0	0	0	0	1	1	1	0	1	1	1	0
0	0	0	0	1	0	0	0	0	0	0	0	1	1	1	0	1	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	1	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	1	0	1
0	0	0	0	0	0	0	1	0	0	0	1	0	1	1	1	0	1	1
0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	1	1	0
0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	0
															0	1	0	0
															0	0	1	0
															0	0	0	1

FIGURE 5: Generator matrix and parity check matrix for the Hamming code of length 15.

Table 1: Parameters of each of the codes discussed and those of some related codes.

	No parity check bit					With parity check bit			
	k	n	d	r	δ	n	d	r	δ
Strength 1 codes:									
Repetition code	1	3	3	0.33	1	4	4	0.25	1
Hamming codes	4	7	3	0.57	0.43	8	4	0.50	0.5
	11	15	3	0.73	0.2	16	4	0.69	0.25
	26	31	3	0.84	0.1	32	4	0.81	0.13
Strength 3 codes:									
Repetition code	1	7	7	0.14	1	8	8	0.13	1
RM(1, 4)	5	15	7	0.33	0.47	16	8	0.31	0.5
RM(2, 5)	16	31	7	0.52	0.23	32	8	0.50	0.25
Golay code	12	23	7	0.52	0.3	24	8	0.50	0.33
Strength 7 codes:									
Repetition code	1	15	15	0.07	1	16	16	0.06	1
RM(1, 5)	6	31	15	0.19	0.48	32	16	0.19	0.5
RM(2, 6)	22	63	15	0.35	0.24	64	16	0.34	0.25
RM(3, 7)	64	127	15	0.50	0.12	128	16	0.50	0.13

A linear code is described by giving three parameters, the length n, the dimension k and the minimum Hamming distance d, in the format $[n, k, d]$. In this notation, the Hamming code described above is a [7, 4, 3] code, and has rate $4/7 = 0.57$, a significant improvement upon the [3, 1, 3] triple repetition code.

Notice that if errors occur in *two* bits, then the resulting sequence will be incorrectly decoded. This can be remedied by appending an additional parity check bit to each code word. This is easily implemented by appending a parity check bit to each row of the generator matrix, as shown in **Figure 4**.

The resulting code is an [8, 4, 4] code, and is called an *extended Hamming code*. Errors in two bits are detected, but cannot be corrected. This improvement comes at the expense of a slightly reduced rate of $^4/_8 = 0.50$.

The Hamming code that we have described is the first in a sequence of Hamming codes of increasing length. For each positive integer m greater than 2, there is a Hamming code of length 2^m-1, dimension 2^m-1-m and minimum distance 3. The case $m = 2$ is the triple repetition code, and the case $m = 3$ is the Hamming code of length 7. For $m = 4$ we get a Hamming code with parameters [15, 11, 3]. The generator matrix and parity check matrix for this code are shown in **Figure 5**.

Because the minimum distance is 3, this code can also correct an error in a single bit. The rate is $^{11}/_{15} = 0.73$, an improvement upon the [7, 4, 3] Hamming code. For $m = 5$ we get a Hamming code with parameters [31, 26, 3], and an even higher rate of $^{26}/_{31} = 0.84$. However, as the length of the code words increases it become more probable that two or more errors will occur within the same block, leading to incorrect decoding.

As with the Hamming code of length 7, an additional parity check bit may be appended to each of these codes to give [16, 11, 4] and [32, 26, 4] extended Hamming codes.

The Hamming code of length n has the following remarkable property: every binary sequence of length n is either a code word, or has a Hamming distance of one between it and an unique code word. A code having this property is called a *perfect code*.

A code that can correct a single error in each block is said to have *strength* 1. If signals are weak or interference is present then it may be expected that more than one error occurs per block, rendering a code of strength 1 useless. For this reason, more robust codes have been developed that are capable of correcting multiple errors in each block.

Multiple error correction – first order Reed-Muller codes

We have already seen that the triple repetition code can correct a single error. A repetition code of length 5 can detect up to two errors per block, and a repetition code of length 7 can detect up to three errors per block. However, the rates of $^1/_5 = 0.20$ and $^1/_7 = 0.14$ respectively are very low.

1	1	1	1	1	1	1	1
1	1	1	1	0	0	0	0
1	1	0	0	1	1	0	0
1	0	1	0	1	0	1	0

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

FIGURE 6: Generator matrices for the Reed-Muller codes RM(1, 3) and RM(1, 4).

The first order Reed-Muller codes RM(1, 3) and RM(1, 4) have generator matrices shown in **Figure 6**. These are named after Irving S Reed and David E Muller. Hopefully the pattern in these will be clear to the reader. RM(1, 3) is an [8, 4, 4] code, and turns out to be equivalent to the extended Hamming code.

In RM(1, 4) each non-zero code word is of weight 8 or 16. It follows that the minimum Hamming distance is 8, and so RM(1, 4) is a [16, 5, 8] code. Because $d = 8$ the code can correct up to three errors per block, and can also detect (but not correct) a fourth error. We have seen that a Hamming code may be extended by appending a parity check bit to each code word; conversely, a Reed-Muller code may be *punctured* by deleting the final bit in each code word. Doing this with RM(1, 4) reduces d by 1 to give a [15, 5, 7] code, which also corrects up to three errors per block and but has a marginally higher rate. These codes are said to have strength 3.

More generally, for each positive integer m, the first order Reed-Muller code RM(1, m) has length 2^m, dimension $1 + m$, and minimum distance 2^{m-1}, and hence is a $[2^m, 1 + m, 2^{m-1}]$ code.

For example, RM(1, 5) is a [32, 6, 16] code, and is able to correct up to seven errors per block. This code was employed by the Mariner spacecraft in the early 1970s to transmit images of Mars back to Earth.

Second order Reed-Muller codes and Golay codes

The generator matrix for the second order Reed-Muller code RM(2, 4) may be obtained from the matrix for RM(1, 4) by the following procedure: consider the four rows that correspond to code words of weight eight (all but the top row). For each pair of such rows, obtain a new row by multiplying the two rows together bit by bit. For example,
1111111100000000 x 1100110011001100 = 1100110000000000

There are six pairs of such rows, and hence the dimension of the new code is 11. The generator matrix for RM(2, 4) appears in **Figure 7**.

RM(2, 4) is a [16, 11, 4] code, and turns out to be equivalent to the extended Hamming code of the same length. But RM(2, 5) is a new code with parameters [32, 16, 8] that, like RM(1, 4), is capable of correcting up to three errors per block but has a higher rate than RM(1, 4).

More generally, RM(2, m) has parameters $[2^m, 1 + \frac{1}{2}m(m+1), 2^{m-2}]$. For example, RM(2, 6) has parameters [64, 22, 16] and is able to correct up to seven errors per block.

Higher order Reed-Muller codes may also be constructed. For example, to construct the third order Reed-Muller code RM(3, m) we include additional rows in the matrix corresponding to the result of multiplying together any two rows or any three rows (excluding the top row) of the generator matrix of RM(1, m).

Finally, we mention the binary Golay code, discovered by Marcel J E Golay in 1949. This is a [23, 12, 7] code. Remarkably, this code achieves the same error correction capability as the punctured [31, 16, 7] Reed-Muller RM(2, 5) code, with marginally higher rate but significantly shorter block length. As with the Hamming codes, the Golay code may be extended by appending a parity check bit to each code word to give a [24, 12, 8] code. The generator matrix for the extended Golay code appears in **Figure 8**.

This code was used by the *Voyager 1* and *Voyager 2* spacecraft to transmit colour pictures of Jupiter and Saturn back to Earth from 1979 through to 1981.

The Golay code of length 23 is also a perfect code: every 23 bit sequence is either a code word, or has a Hamming distance of three or less between it and a unique code word. It has been proved that any perfect binary linear code has the parameters of the Golay code or a (non-extended) Hamming code – there are no other cases.

There are other binary linear codes that are beyond the scope of this article, notably the BCH codes, named after the French mathematician Alexis Hocquenghem, who discovered these in 1959, and Raj Bose and D K Ray-Chaudhuri who discovered them independently in 1960. The BCH codes achieve a better rate than Reed-Muller codes of comparable minimum Hamming distance and length. Codes may also be defined over larger alphabets. For example, a *ternary* code has an alphabet with three symbols: 0, 1, 2. Alphabets of size a power of two are particularly useful. For example, the hexadecimal alphabet comprises sixteen symbols – numerals 0 through 9 and letters A through F. Each hexadecimal symbol is equivalent to four bits, so that the bit rate for a hexadecimal code is four times the symbol rate. The Hamming codes and Reed-Muller codes can be constructed over alphabets of different sizes. There are also powerful codes constructed over larger alphabets that are not applicable in the binary case, notably the Reed-Solomon codes. These are used in compact discs, and also in the FX25 packet protocol.

Table 1 summarises the parameters of each of the codes that we have discussed, and those of some related codes. For each code, the table gives the dimension k, the length n, and the minimum Hamming distance d. The rate r is calculated as k/n, and the relative distance δ is calculated as d/n. Notice the trade-off between rate and relative distance: as r increases, δ tends to decrease, and vice-versa. Values of n, d, r and δ are given for codes with and without an additional parity check bit.

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0

FIGURE 7: The generator matrix for RM(2, 4)

1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	0	1	0	
0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	0	1	
0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1	1	1	1	0
0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	1	1	1	0	1
0	0	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	1	0	0	1	1	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	0	0	1	0	0	1	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1	1	0	0	1	0	0	1	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1	1	0	0	1	0	0	1
0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1	1	0	0	1	1	0
0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	0	1	0	1	1	1
0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	1	0	1	0	1	1

FIGURE 8: The generator matrix for the extended Golay code.